Tập Đoàn Công Nghiệp – Viễn Thông Quân Đội (Viettel) Viettel Digital Talent Program



Tài Liệu Báo Cáo Mini Project

Hệ Thống Điều Khiển

Phiên Họp Trực Tuyến

Người thực hiện: Ngô Minh Đức

Lĩnh vực: Software Engineering (Web)

Mentor: Lê Trung Hiếu (VTS)

*Hanoi, Tháng 6 2025*

**Mục lục**

[1. Giới thiệu 3](#_Toc200563398)

[1.1. Bài toán 3](#_Toc200563399)

[1.2. Phạm vi và mục tiêu 4](#_Toc200563400)

[2. Giải pháp 5](#_Toc200563401)

[2.1. Tổng quan giải pháp 5](#_Toc200563402)

[2.1.1. Mô hình hoạt động 5](#_Toc200563403)

[2.1.2. Mô tả luồng hoạt động 5](#_Toc200563404)

[2.2. Triển khai backend Springboot và scale theo chiều ngang 6](#_Toc200563405)

[2.2.1. Horizontal Scaling 6](#_Toc200563406)

[2.3. Triển khai Load Balancer với Nginx 7](#_Toc200563407)

[2.3.1. Tại sao cần load balancing 7](#_Toc200563408)

[2.3.2. Triển khai Nginx để cân bằng tải 7](#_Toc200563409)

[2.4. Sử dụng Kafka để giao tiếp giữa các instances 7](#_Toc200563410)

[2.4.1. Vấn đề đặt ra 7](#_Toc200563411)

[2.4.2. Triển khai Kafka 7](#_Toc200563412)

[2.5. Xử lí realtime bằng WebSocket 8](#_Toc200563413)

[2.5.1. Các phương pháp xử lí realtime 8](#_Toc200563414)

[2.5.2. Triển khai Websocket 9](#_Toc200563415)

[2.6. Đóng gói Docker 9](#_Toc200563416)

[3. Load Testing 9](#_Toc200563417)

[3.1. Phương pháp đánh giá 9](#_Toc200563418)

[3.2. Kết quả đánh giá 9](#_Toc200563419)

[4. Sản phẩm và thử nghiệm 9](#_Toc200563420)

[5. Tài liệu tham khảo 9](#_Toc200563421)

[6. Phụ lục 9](#_Toc200563422)

# Giới thiệu

## Bài toán

Trong bối cảnh các cuộc họp, hội thảo và phiên họp chuyển dần sang hình thức trực tuyến, nhu cầu về một hệ thống điều khiển phiên họp mạnh mẽ, ổn định và hỗ trợ thao tác theo thời gian thực trở nên thiết yếu. Bài toán đặt ra là xây dựng một hệ thống điều khiển phiên họp trực tuyến dựa trên kiến trúc microservices, có khả năng phục vụ đồng thời số lượng lớn người dùng mà vẫn đảm bảo tính nhất quán và độ trễ thấp trong việc xử lý các hành động.

Về mặt chức năng, hệ thống cuộc họp này cần cung cấp các tính năng quan trọng như: biểu quyết trực tuyến (vote), và đăng ký phát biểu trong phiên họp. Các chức năng này cần phải được lưu lại lịch sử. Chúng phải hoạt động ổn định, đồng bộ và đáp ứng ngay lập tức theo thời gian thực nhằm tạo trải nghiệm liền mạch cho người tham gia.

Hệ thống này cũng cần đáp ứng được một vài yêu cầu kĩ thuật. Cụ thể, nó phải hỗ trợ tương tác real-time với hệ thống, kể cả khi số lượng người tham gia lớn. Người dùng khi tải lại trang (reload) hoặc bị chuyển sang một instance khác của hệ thống (ví dụ do load balancing) thì vẫn phải đồng bộ được với các sự kiện đang diễn ra mà không bị gián đoạn. Ngoài ra, việc lưu trữ thứ tự và lịch sử thao tác là rất quan trọng để đảm bảo tính chính xác và minh bạch trong các hoạt động như biểu quyết hoặc đăng ký phát biểu.

## Phạm vi và mục tiêu

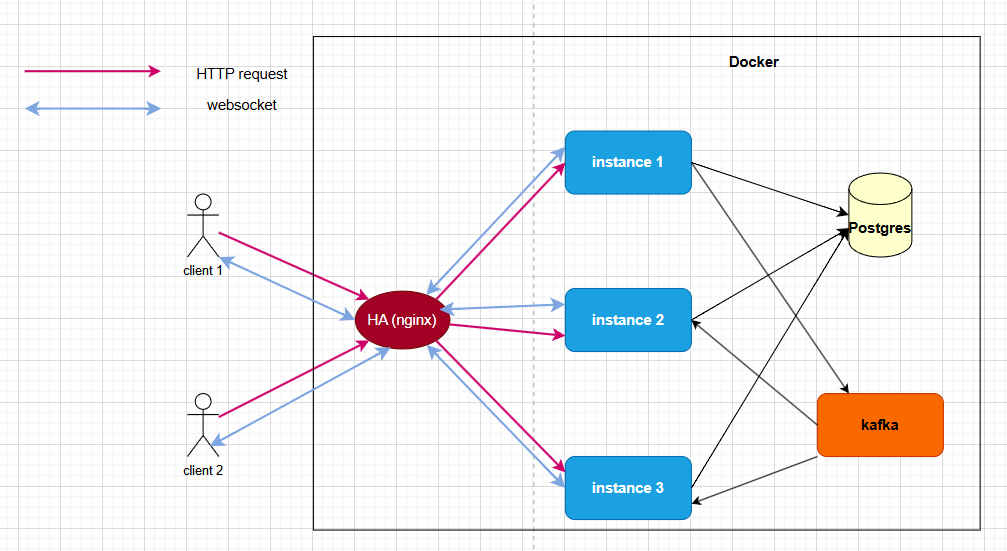
Trong phạm vi của Mini Project, hệ thống được xây dựng, triển khai và thử nghiệm sẽ bao gồm:

* Nginx làm reverse proxy với nhiệm vụ cân bằng tải
* Server backend (sử dụng Springboot) được scale theo chiều ngang thành nhiều instances.
* Database sử dụng PostgreSQL
* Apache Kafka để giao tiếp giữa các backend instance
* Hệ thống được đóng gói bằng Docker

# Giải pháp

## Tổng quan giải pháp

### Mô hình hoạt động



Hình 1: Mô hình hoạt động

### Mô tả luồng hoạt động

Backend server sẽ được scale thành nhiều instance.

Khi người dùng (client) thực hiện một hành động như biểu quyết hoặc đăng ký phát biểu, hệ thống sẽ tiếp nhận yêu cầu thông qua một HTTP POST request. Yêu cầu này được gửi đến máy chủ thông qua Nginx, trước khi được chuyển tiếp đến một instance xử lý.

Tại instance này, request sẽ được xử lý và lưu thời điểm người dùng hành động vào trong database Postgres. Do đó, lịch sử tác động của người dùng sẽ được lưu lại. Đồng thời, thông tin về loại request (biểu quyết hay đăng kí phát biểu), cùng với ID của cuộc họp sẽ được đóng gói thành một message và gửi vào một **Kafka Topic**. Mỗi cuộc họp được ánh xạ với một topic riêng biệt trên Kafka nhằm đảm bảo sự phân tách và quản lý độc lập giữa các phiên họp.

Các instance khác trong hệ thống, vốn đã subscribe (đăng ký lắng nghe) topic tương ứng, sẽ nhận được thông điệp mới khi nó được phát tán (publish). Sau khi nhận được message, hệ thống sẽ sử dụng WebSocket để đẩy thông tin này về phía client theo thời gian thực.

Sau khi nhận được phản hồi Websocket, client sẽ gọi HTTP GET request để cập nhật lại toàn bộ trạng thái dữ liệu của phiên họp (như danh sách người phát biểu, kết quả biểu quyết, v.v.). Đây là phương pháp thực thi event notification.

## Triển khai backend Springboot và scale theo chiều ngang

### Horizontal Scaling

Trong kiến trúc hệ thống hiện đại, khả năng mở rộng (scaling) là yếu tố then chốt để đảm bảo hiệu suất và độ tin cậy. Có hai phương pháp scaling chính được sử dụng phổ biến: Horizontal Scaling (mở rộng ngang) và Vertical Scaling (mở rộng dọc).

Vertical Scaling là phương pháp nâng cấp cấu hình phần cứng của một máy chủ duy nhất để tăng khả năng xử lý. Cách tiếp cận này bao gồm việc bổ sung thêm CPU, RAM, hoặc nâng cấp ổ cứng cho server hiện có. Ưu điểm chính của Vertical Scaling là tính đơn giản trong triển khai và bảo trì, do không yêu cầu thay đổi kiến trúc hệ thống hay phát sinh các vấn đề về đồng bộ dữ liệu.

Tuy nhiên, Vertical Scaling tồn tại một số hạn chế đáng kể. Về mặt vật lý, khả năng nâng cấp phần cứng bị giới hạn bởi thông số kỹ thuật tối đa của máy chủ. Khi đạt đến ngưỡng này, hệ thống không thể mở rộng thêm. Ngoài ra, phương pháp này dẫn đến Single Point Failure (1 máy chủ hỏng hóc sẽ gây ra sự cố cho toàn hệ thống), vì toàn bộ hệ thống phụ thuộc vào một máy chủ duy nhất. Chi phí nâng cấp phần cứng cũng thường cao hơn so với việc đầu tư thêm nhiều máy chủ có cấu hình thấp hơn.

Do những hạn chế của Vertical Scaling kể trên, cũng như những khó khăn trong việc triển khai thử nghiệm local, mini project sử dụng Horizontal Scaling.

Horizontal Scaling là phương pháp tăng khả năng xử lý bằng cách thêm nhiều máy chủ vào hệ thống và phân phối tải giữa chúng. Cách tiếp cận này yêu cầu sử dụng các công nghệ như cân bằng tải (load balancing) và thường đi kèm với kiến trúc phân tán. Ưu điểm nổi bật của Horizontal Scaling là khả năng mở rộng gần như vô hạn, chỉ cần tiếp tục bổ sung thêm máy chủ khi nhu cầu tăng. Phương pháp này cũng giúp loại bỏ Single Point Failure, nâng cao tính sẵn sàng của hệ thống (High Avaibility).

## Triển khai Load Balancer với Nginx

### Tại sao cần load balancing

Load Balancing (Cân bằng tải) là kỹ thuật phân phối đều lưu lượng truy cập hoặc yêu cầu từ người dùng đến nhiều máy chủ (server/instance) trong một hệ thống. Mục tiêu của load balancing là đảm bảo rằng không có máy chủ nào bị quá tải, từ đó giúp hệ thống hoạt động hiệu quả, ổn định và có khả năng mở rộng.

Trong bối cảnh hệ thống điều khiển phiên họp trực tuyến, nơi có thể có hàng trăm hoặc hàng nghìn người dùng tham gia cùng lúc, load balancing đảm bảo rằng không một instance nào bị quá tải, từ đó duy trì hiệu suất ổn định và khả năng đáp ứng theo thời gian thực. Ngoài ra, load balancing giúp tăng tính sẵn sàng của hệ thống bằng cách chuyển hướng lưu lượng đến các instance còn hoạt động khi một instance gặp sự cố, từ đó giảm thiểu thời gian gián đoạn và nâng cao trải nghiệm người dùng.

### Triển khai Nginx để cân bằng tải

## Sử dụng Kafka để giao tiếp giữa các instances

### Vấn đề đặt ra

Trong hệ thống điều khiển phiên họp trực tuyến, yêu cầu về cập nhật theo thời gian thực, xử lý đồng thời số lượng lớn người dùng, và đồng bộ trạng thái giữa các instance là rất quan trọng. Để đáp ứng các yêu cầu này, hệ thống sử dụng Apache Kafka làm nền tảng giao tiếp chính giữa các instance.

Apache Kafka là một nền tảng message queue phân tán, được thiết kế để xử lý dòng dữ liệu lớn, tốc độ cao và tin cậy, rất phù hợp cho các hệ thống microservices cần trao đổi dữ liệu một cách liên tục và không đồng bộ.

### Triển khai Kafka

# Xử lí realtime bằng WebSocket

## Các phương pháp xử lí realtime

Trong các hệ thống yêu cầu cập nhật dữ liệu theo thời gian thực, như hệ thống điều khiển phiên họp trực tuyến, việc lựa chọn phương pháp truyền thông realtime là yếu tố then chốt để đảm bảo trải nghiệm người dùng mượt mà và đáng tin cậy. Có ba kỹ thuật phổ biến được sử dụng là HTTP Polling, WebSocket và Server-Sent Events (SSE). Mỗi kỹ thuật có ưu điểm và hạn chế riêng, tùy thuộc vào đặc thù ứng dụng và kiến trúc hệ thống.

HTTP Polling là phương pháp đơn giản và phổ biến nhất. Trong mô hình này, client sẽ gửi các yêu cầu định kỳ (thường là HTTP GET) đến server để kiểm tra xem có dữ liệu mới không. Nếu có, server phản hồi dữ liệu; nếu không, phản hồi sẽ rỗng hoặc không đổi. Dù dễ triển khai và tương thích tốt với hầu hết nền tảng, phương pháp này tiêu tốn nhiều tài nguyên vì phải gửi request liên tục, kể cả khi không có thay đổi. Do đó, độ trễ cao và hiệu suất thấp là những nhược điểm chính khiến HTTP Polling không phù hợp với ứng dụng cần phản hồi tức thì.

WebSocket là một giao thức nâng cao hơn, cho phép thiết lập kết nối hai chiều (full-duplex) giữa client và server. Sau khi hoàn tất bắt tay ban đầu bằng HTTP, kết nối sẽ được duy trì liên tục và cho phép trao đổi dữ liệu thời gian thực giữa hai bên mà không cần thiết lập lại. WebSocket đặc biệt hiệu quả cho các ứng dụng cần truyền dữ liệu liên tục, tương tác thời gian thực như chat, họp trực tuyến hoặc trò chơi đa người. Tuy nhiên, nó yêu cầu cấu hình máy chủ và hạ tầng mạng phức tạp hơn, đặc biệt là khi làm việc với proxy, firewall hoặc load balancer.

Server-Sent Events (SSE), còn gọi là Event Stream, là một phương pháp khác để truyền dữ liệu realtime từ server về client. Trong mô hình này, kết nối HTTP được mở một chiều và giữ nguyên để server có thể “đẩy” dữ liệu đến client mỗi khi có sự kiện mới xảy ra. SSE đơn giản hơn WebSocket trong trường hợp chỉ cần server → client (một chiều), như trong việc hiển thị kết quả biểu quyết hoặc cập nhật danh sách người phát biểu. SSE hoạt động tốt với HTTP/1.1, có cơ chế tự động reconnect và tương thích với hầu hết trình duyệt hiện đại (trừ một số hạn chế trên Internet Explorer). Tuy nhiên, nếu ứng dụng cần tương tác hai chiều thì WebSocket sẽ phù hợp hơn.

Hệ thống phiên họp trực tuyến yêu cầu khả năng cập nhật dữ liệu theo thời gian thực và phản hồi tức thì giữa người dùng và máy chủ. WebSocket đáp ứng tốt nhu cầu này nhờ hỗ trợ kết nối hai chiều liên tục, giúp server có thể đẩy dữ liệu đến client ngay khi có sự kiện mới mà không cần polling liên tục. So với các phương pháp khác như HTTP Polling (độ trễ cao) hoặc SSE (chỉ truyền một chiều), WebSocket cho hiệu năng cao hơn, độ trễ thấp hơn và phù hợp với môi trường có nhiều người dùng đồng thời. Vì vậy, mini project sử dụng Websocket.

## Triển khai Websocket

# Đóng gói Docker

# Load Testing

## Phương pháp đánh giá

## Kết quả đánh giá

# Sản phẩm và thử nghiệm

# Tài liệu tham khảo

# Phụ lục